# Quick Sort

## Splitting

The logic of splitting process is explained as follows:

1. Pick partitioning element `p = a[i]`.
2. Set two cursors, `i = 1` and `j = n-1`.
3. If `a[i]` $<$ `p` increment `i`
4. If `a[j]` $>$ `p` decrement `j`
5. If `i` $<=$ `j` exchange `a[i]` and `a[j]`
6. When `i` crosses `j` we are in boundary of smaller half
7. So, `p` must be placed there.

# Quick Sort

## Quick Sort (Splitting)

```
int split(int a[], int i, int j) {
    int x = a[i]; // splitting element

    while (1) {
        while (i < j && x <= a[j]) j--; // decrement until a[j] > x
        if (i >= j) break;  // check if i meets/crosses j
        a[i++] = a[j];
        while (i < j && x >= a[i]) i++;
        if (i >= j) break;
        a[j--] = a[i];
    }

    a[j] = x;  // place x in a[j]  when i meets/crosses j
    return j;  // return partition index.
}
```

# Quick Sort

## Quick Sort (Recursion)

```
void quickSort(int a[], int l, int h) {
    int m;
    if (l >= h)
        return;
    m = split(a, l, h);
    quickSort(a, l, m - 1);
    quickSort(a, m + 1, h);
}
```

# Quick Sort

## Putting All Together

```
#include <stdio.h>
#define N 10
int main() {
    int a[N], i;
    int n = sizeof(a)/sizeof(a[0]);

    generate(a, n);
    printf("Unsorted numbers\n");
    printArray(a, n);

    quickSort(a, 0, n - 1);
    printf("Sorted numbers\n");
    printArray(a, n);
}
```

# Pointers

## Have We Seen?

- Consider when a variable is read using `scanf`.
- Operator `&` gives location where `scanf` should store value.
- This address is mapped to the name of the variable being read.
- Reading/accessing variable, would thus provide the input value.
- So, we are already familiar with pointer.

# Pointers

## Why We Need Pointers?

- Consider exchanging values of two variables.
- With automatic variables swapping is not possible.
- On return from swap function, values remain unaltered.
- Swapping values of global variables are visible.
- But use of global variables lead to uncontrolled side-effects, making programs hard to understand.
- Pointers provide mechanism to implement swap like functions.
- They make programs efficient and compact.